TensorFlow与中文手写汉字识别

2017/03/20 · 实践项目 · 3 评论 · tensorflow, 汉字识别

 原文出处： 小石头（@小石头_sh ）

Goal

本文目标是利用TensorFlow做一个简单的图像分类器，在比较大的数据集上，尽可能高效地做图像相关处理，从Train，Validation到Inference，是一个比较基本的Example， 从一个基本的任务学习如果在TensorFlow下做高效地图像读取，基本的图像处理，整个项目很简单，但其中有一些trick，在实际项目当中有很大的好处， 比如绝对不要一次读入所有的 的数据到内存（尽管在Mnist这类级别的例子上经常出现)...

最开始看到是这篇blog里面的TensorFlow练习22: 手写汉字识别, 但是这篇文章只用了140训练与测试，试了下代码 很快，但是当扩展到所有的时，发现32g的内存都不够用，这才注意到原文中都是用numpy，会先把所有的数据放入到内存，但这个不必须的，无论在MXNet还是TensorFlow中都是不必须的，MXNet使用的是DataIter，会在程序运行的过程中异步读取数据，TensorFlow也是这样的，TensorFlow封装了高级的api，用来做数据的读取，比如TFRecord，还有就是从filenames中读取， 来异步读取文件，然后做shuffle batch，再feed到模型的Graph中来做模型参数的更新。具体在tf如何做数据的读取可以看看reading data in tensorflow

这里我会拿到所有的数据集来做训练与测试，算作是对斗大的熊猫上面那篇文章的一个扩展。

Batch Generate

数据集来自于中科院自动化研究所，感谢分享精神！！！具体下载:

```
wget http://www.nlpr.ia.ac.cn/databases/download/feature_data/HWDB1.1trn_gnt.zip
wget http://www.nlpr.ia.ac.cn/databases/download/feature_data/HWDB1.1tst_gnt.zip

```

解压后发现是一些gnt文件，然后用了斗大的熊猫里面的代码，将所有文件都转化为对应label目录下的所有png的图片。（注意在HWDB1.1trn_gnt.zip解压后是alz文件，需要再次解压 我在mac没有找到合适的工具，windows上有alz的解压工具）。

Python

```
import os
import numpy as np
import struct
from PIL import Image


data_dir = '../data'
train_data_dir = os.path.join(data_dir, 'HWDB1.1trn_gnt')
test_data_dir = os.path.join(data_dir, 'HWDB1.1tst_gnt')


def read_from_gnt_dir(gnt_dir=train_data_dir):
    def one_file(f):
```

```
            header = np.fromfile(f, dtype='uint8', count=header_size)
            if not header.size: break
            sample_size = header[0] + (header[1]<<8) + (header[2]<<16) + (header[3]<<24)
            tagcode = header[5] + (header[4]<<8)
            width = header[6] + (header[7]<<8)
            height = header[8] + (header[9]<<8)
            if header_size + width*height != sample_size:
                break
            image = np.fromfile(f, dtype='uint8', count=width*height).reshape((height, width))
            yield image, tagcode
    for file_name in os.listdir(gnt_dir):
        if file_name.endswith('.gnt'):
            file_path = os.path.join(gnt_dir, file_name)
            with open(file_path, 'rb') as f:
                for image, tagcode in one_file(f):
                    yield image, tagcode
char_set = set()
for _, tagcode in read_from_gnt_dir(gnt_dir=train_data_dir):
    tagcode_unicode = struct.pack('>H', tagcode).decode('gb2312')
    char_set.add(tagcode_unicode)
char_list = list(char_set)
char_dict = dict(zip(sorted(char_list), range(len(char_list))))
print len(char_dict)
import pickle
f = open('char_dict', 'wb')
pickle.dump(char_dict, f)
f.close()
train_counter = 0
test_counter = 0
for image, tagcode in read_from_gnt_dir(gnt_dir=train_data_dir):
    tagcode_unicode = struct.pack('>H', tagcode).decode('gb2312')
    im = Image.fromarray(image)
    dir_name = '../data/train/' + '%0.5d'%char_dict[tagcode_unicode]
    if not os.path.exists(dir_name):
        os.mkdir(dir_name)
    im.convert('RGB').save(dir_name+'/' + str(train_counter) + '.png')
    train_counter += 1
for image, tagcode in read_from_gnt_dir(gnt_dir=test_data_dir):
    tagcode_unicode = struct.pack('>H', tagcode).decode('gb2312')
    im = Image.fromarray(image)
    dir_name = '../data/test/' + '%0.5d'%char_dict[tagcode_unicode]
    if not os.path.exists(dir_name):
        os.mkdir(dir_name)
    im.convert('RGB').save(dir_name+'/' + str(test_counter) + '.png')
    test_counter += 1
```

处理好的数据，放到了云盘，大家可以直接在我的云盘来下载处理好的数据集HWDB1. 这里说明下，char_dict是汉字和对应的数字label的记录。

得到数据集后，就要考虑如何读取了，一次用numpy读入内存在很多小数据集上是可以行的，但是在稍微大点的数据集上内存就成了瓶颈，但是不要害怕，TensorFlow有自己的方法：

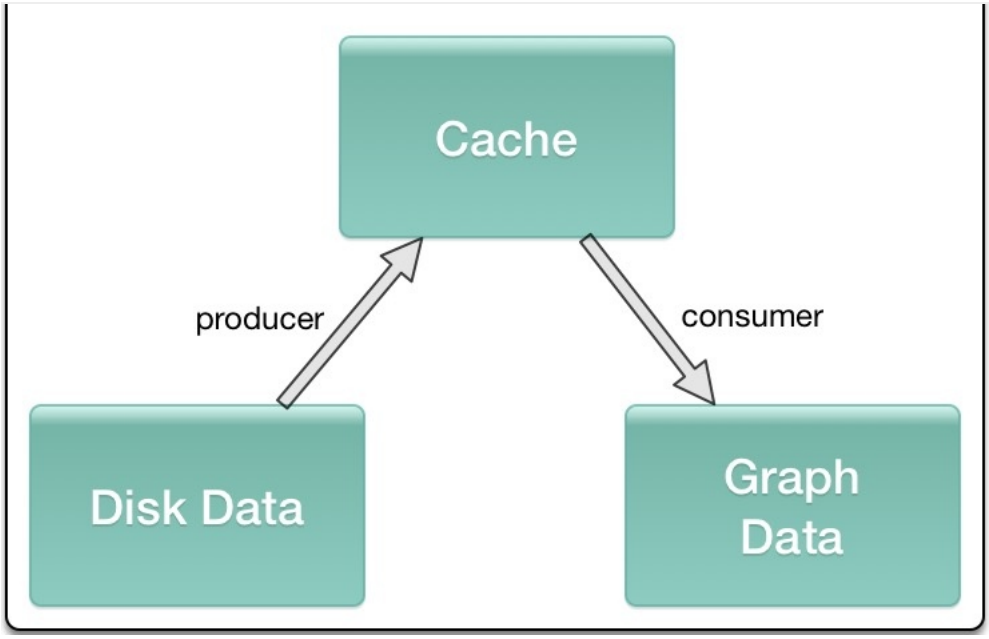
```
def batch_data(file_labels,sess, batch_size=128):
    image_list = [file_label[0] for file_label in file_labels]
    label_list = [int(file_label[1]) for file_label in file_labels]
    print 'tag2 {0}'.format(len(image_list))
    images_tensor = tf.convert_to_tensor(image_list, dtype=tf.string)
    labels_tensor = tf.convert_to_tensor(label_list, dtype=tf.int64)
    input_queue = tf.train.slice_input_producer([images_tensor, labels_tensor])

    labels = input_queue[1]
    images_content = tf.read_file(input_queue[0])
    # images = tf.image.decode_png(images_content, channels=1)
    images = tf.image.convert_image_dtype(tf.image.decode_png(images_content, channels=1), tf.float32)
    # images = images / 256
    images =  pre_process(images)
    # print images.get_shape()
    # one hot
    labels = tf.one_hot(labels, 3755)
    image_batch, label_batch = tf.train.shuffle_batch([images, labels], batch_size=batch_size, capacity=50000,min_after_dequeue=1000
    # print 'image_batch', image_batch.get_shape()

    coord = tf.train.Coordinator()
    threads = tf.train.start_queue_runners(sess=sess, coord=coord)
    return image_batch, label_batch, coord, threads

```

简单介绍下，首先你需要得到所有的图像的path和对应的label的列表，利用`tf.convert_to_tensor`转换为对应的tensor， 利用`tf.train.slice_input_producer`将image_list ,label_list做一个slice处理，然后做图像的读取、预处理，以及label的one_hot表示，然后就是传到`tf.train.shuffle_batch`产生一个个shuffle batch，这些就可以feed到你的 模型。 `slice_input_producer`和`shuffle_batch`这类操作内部都是基于queue，是一种异步的处理方式,会在设备中开辟一段空间用作cache，不同的进程会分别一直往cache中塞数据 和取数据，保证内存或显存的占用以及每一个mini-batch不需要等待，直接可以从cache中获取。



Data Augmentation

由于图像场景不复杂，只是做了一些基本的处理，包括图像翻转，改变下亮度等等，这些在TensorFlow里面有现成的api，所以尽量使用TensorFlow来做相关的处理：

```
def pre_process(images):
    if FLAGS.random_flip_up_down:
        images = tf.image.random_flip_up_down(images)
    if FLAGS.random_flip_left_right:
        images = tf.image.random_flip_left_right(images)
    if FLAGS.random_brightness:
        images = tf.image.random_brightness(images, max_delta=0.3)
    if FLAGS.random_contrast:
        images = tf.image.random_contrast(images, 0.8, 1.2)
    new_size = tf.constant([FLAGS.image_size,FLAGS.image_size], dtype=tf.int32)
    images = tf.image.resize_images(images, new_size)
    return images

```

Build Graph

这里很简单的构造了一个两个卷积+一个全连接层的网络，没有做什么更深的设计，感觉意义不大，设计了一个dict，用来返回后面要用的所有op，还有就是为了方便再训练中查看loss和accuracy， 没有什么特别的，很容易理解, labels 为None时 方便做inference。

```
def network(images, labels=None):
    endpoints = {}
    conv_1 = slim.conv2d(images, 32, [3,3],1, padding='SAME')
    max_pool_1 = slim.max_pool2d(conv_1, [2,2],[2,2], padding='SAME')
    conv_2 = slim.conv2d(max_pool_1, 64, [3,3],padding='SAME')
    max_pool_2 = slim.max_pool2d(conv_2, [2,2],[2,2], padding='SAME')
    flatten = slim.flatten(max_pool_2)
    out = slim.fully_connected(flatten,3755, activation_fn=None)
    global_step = tf.Variable(initial_value=0)
    if labels is not None:
        loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(out, labels))
        train_op = tf.train.AdamOptimizer(learning_rate=0.0001).minimize(loss, global_step=global_step)
        accuracy = tf.reduce_mean(tf.cast(tf.equal(tf.argmax(out, 1), tf.argmax(labels, 1)), tf.float32))
        tf.summary.scalar('loss', loss)
        tf.summary.scalar('accuracy', accuracy)
        merged_summary_op = tf.summary.merge_all()
    output_score = tf.nn.softmax(out)
    predict_val_top3, predict_index_top3 = tf.nn.top_k(output_score, k=3)

    endpoints['global_step'] = global_step
    if labels is not None:
        endpoints['labels'] = labels
        endpoints['train_op'] = train_op
        endpoints['loss'] = loss
        endpoints['accuracy'] = accuracy
        endpoints['merged_summary_op'] = merged_summary_op
    endpoints['output_score'] = output_score
    endpoints['predict_val_top3'] = predict_val_top3
    endpoints['predict_index_top3'] = predict_index_top3
    return endpoints

```

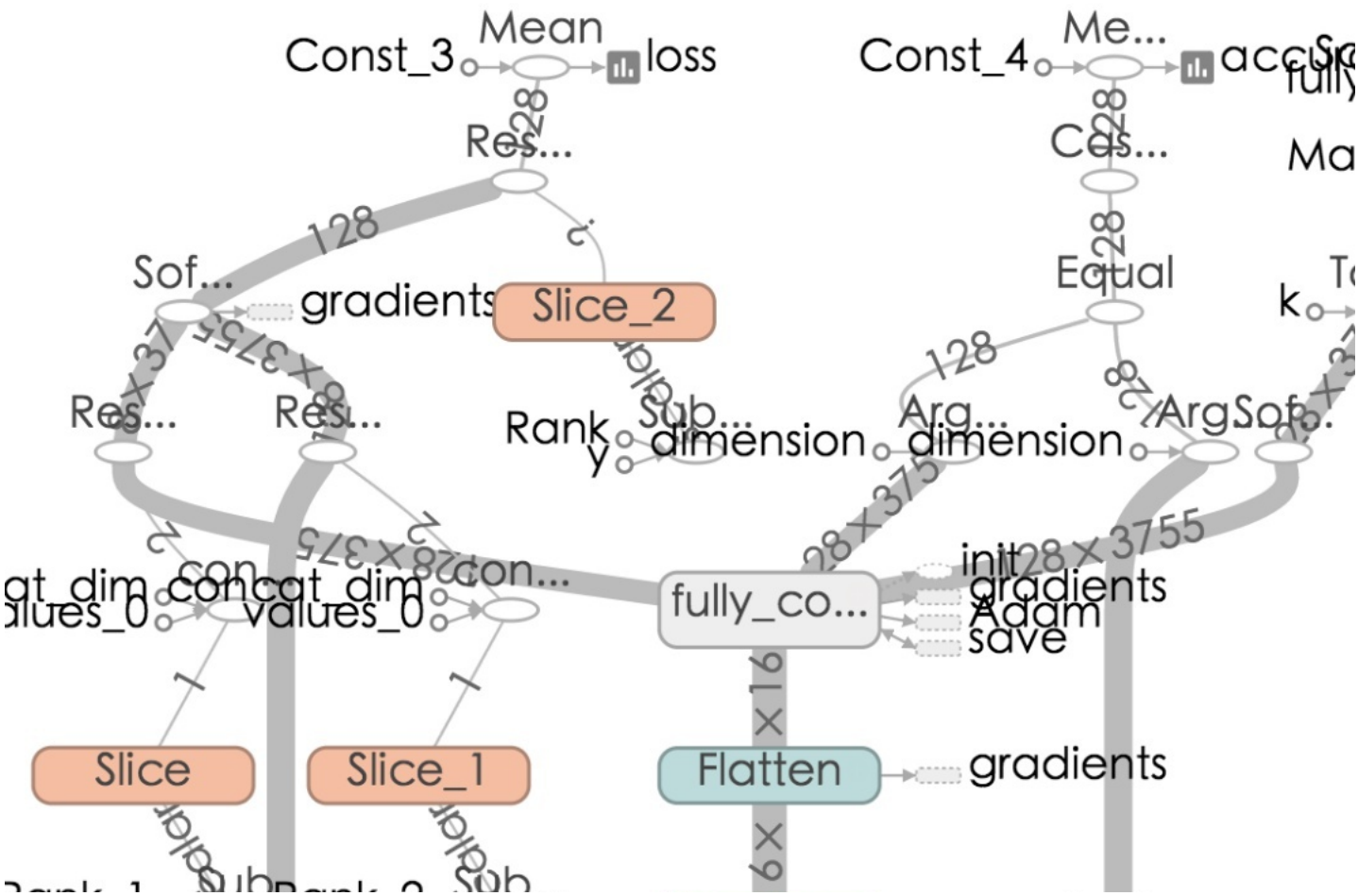
Train

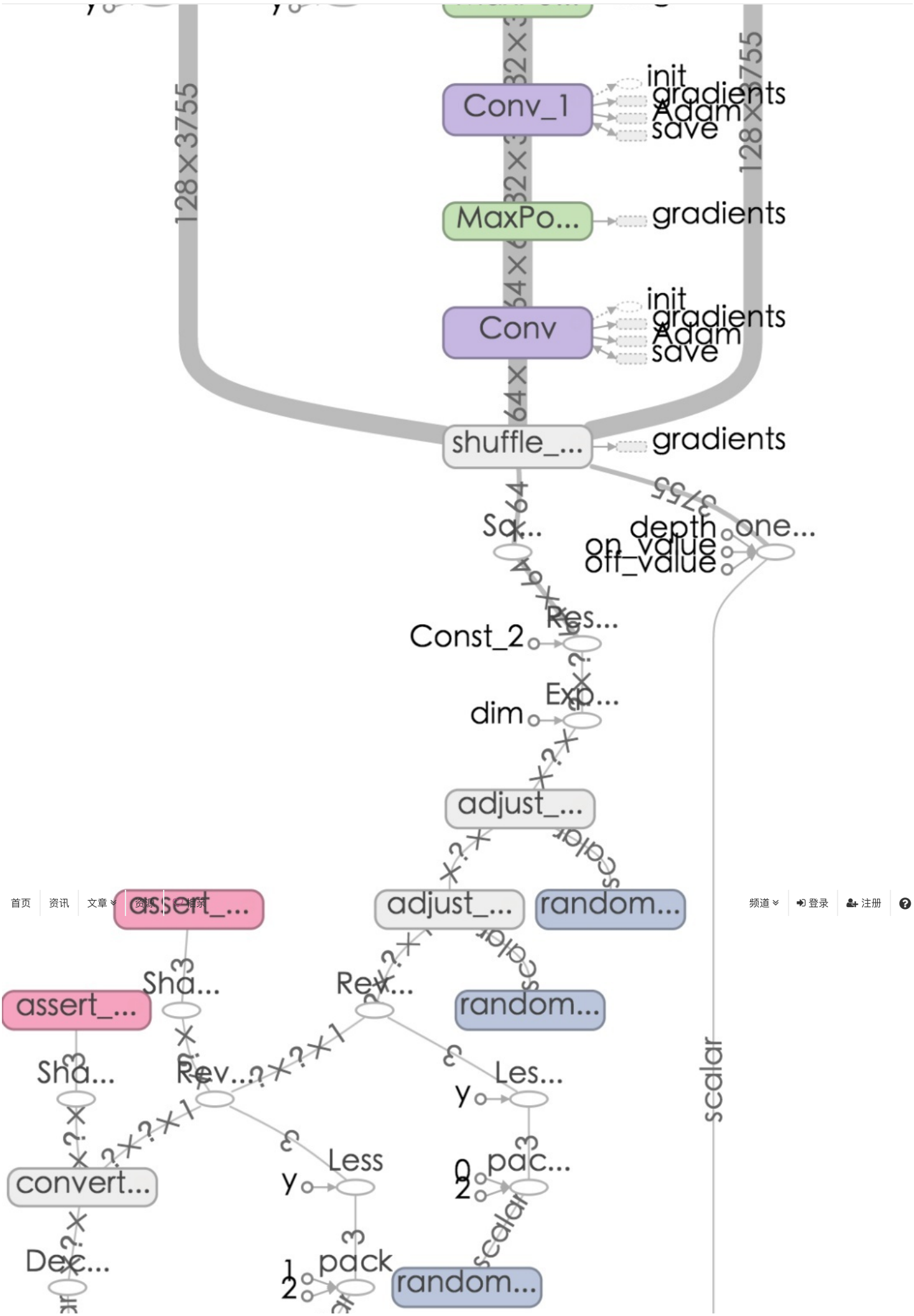
train batch的eval，每隔save_steps 后保存一次checkpoint。

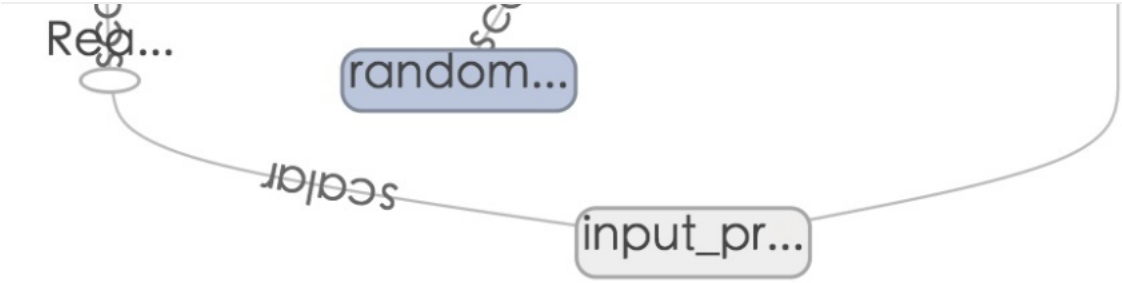
```
def train():
    sess = tf.Session()
    file_labels = get_imagesfile(FLAGS.train_data_dir)
    images, labels, coord, threads = batch_data(file_labels, sess)
    endpoints = network(images, labels)
    saver = tf.train.Saver()
    sess.run(tf.global_variables_initializer())
    train_writer = tf.train.SummaryWriter('./log' + '/train',sess.graph)
    test_writer = tf.train.SummaryWriter('./log' + '/val')
    start_step = 0
    if FLAGS.restore:
        ckpt = tf.train.latest_checkpoint(FLAGS.checkpoint_dir)
        if ckpt:
            saver.restore(sess, ckpt)
            print "restore from the checkpoint {0}".format(ckpt)
            start_step += int(ckpt.split('-')[-1])
    logger.info(':::Training Start:::')
    try:
        while not coord.should_stop():
        # logger.info('step {0} start'.format(i))
            start_time = time.time()
            _, loss_val, train_summary, step = sess.run([endpoints['train_op'], endpoints['loss'], endpoints['merged_summary_op'], e
            train_writer.add_summary(train_summary, step)
            end_time = time.time()
            logger.info("the step {0} takes {1} loss {2}".format(step, end_time-start_time, loss_val))
            if step > FLAGS.max_steps:
                break
            # logger.info("the step {0} takes {1} loss {2}".format(i, end_time-start_time, loss_val))
            if step % FLAGS.eval_steps == 1:
                accuracy_val,test_summary, step = sess.run([endpoints['accuracy'], endpoints['merged_summary_op'], endpoints['global
                test_writer.add_summary(test_summary, step)
                logger.info('===============Eval a batch in Train data=======================')
                logger.info( 'the step {0} accuracy {1}'.format(step, accuracy_val))
                logger.info('===============Eval a batch in Train data=======================')
            if step % FLAGS.save_steps == 1:
                logger.info('Save the ckpt of {0}'.format(step))
                saver.save(sess, os.path.join(FLAGS.checkpoint_dir, 'my-model'), global_step=endpoints['global_step'])
    except tf.errors.OutOfRangeError:
        # print "===========train finished========="
        logger.info('==================Train Finished================')
        saver.save(sess, os.path.join(FLAGS.checkpoint_dir, 'my-model'), global_step=endpoints['global_step'])
    finally:
        coord.request_stop()
    coord.join(threads)
    sess.close()

```

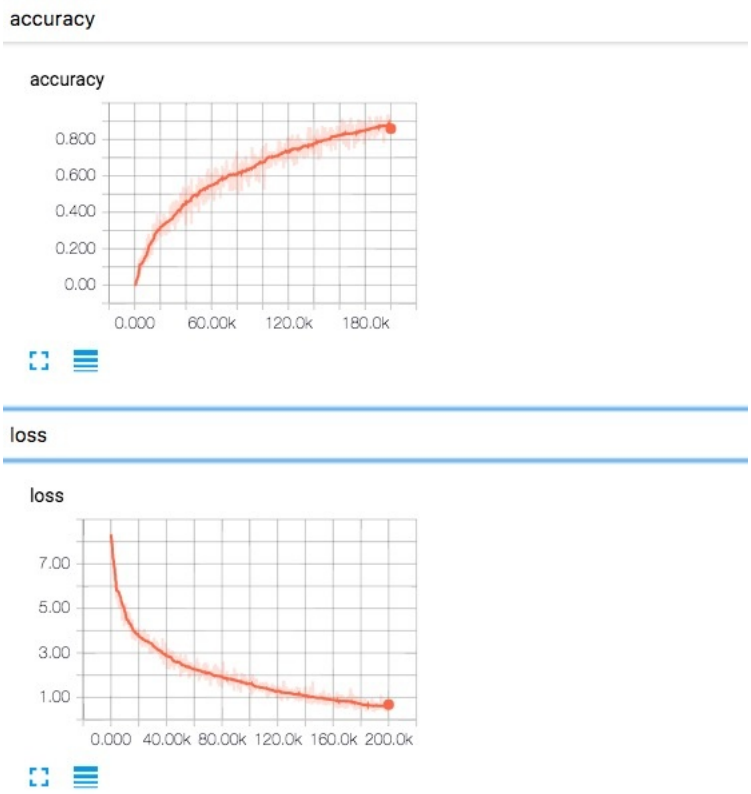
Graph







Loss and Accuracy



Validation

训练完成之后，想对最终的模型在测试数据集上做一个评估，这里我也曾经尝试利用batch_data，将slice_input_producer中epoch设置为1，来做相关的工作，但是发现这里无法和train 共用，会出现epoch无初始化值的问题（train中传epoch为None），所以这里自己写了shuffle batch的逻辑，将测试集的images和labels通过feed_dict传进到网络，得到模型的输出， 然后做相关指标的计算：

```
def validation():
    # it should be fixed by using placeholder with epoch num in train stage
    sess = tf.Session()

    file_labels = get_imagesfile(FLAGS.test_data_dir)
    test_size = len(file_labels)
    print test_size
    val_batch_size = FLAGS.val_batch_size
    test_steps = test_size / val_batch_size
    print test_steps
    # images, labels, coord, threads= batch_data(file_labels, sess)
    images = tf.placeholder(dtype=tf.float32, shape=[None, 64, 64, 1])
    labels = tf.placeholder(dtype=tf.int32, shape=[None,3755])
    # read batch images from file_labels
    # images_batch = np.zeros([128,64,64,1])
    # labels_batch = np.zeros([128,3755])
    # labels_batch[0][20] = 1
    #
    endpoints = network(images, labels)
    saver = tf.train.Saver()
    ckpt = tf.train.latest_checkpoint(FLAGS.checkpoint_dir)
    if ckpt:
        saver.restore(sess, ckpt)
        # logger.info("restore from the checkpoint {0}".format(ckpt))
    # logger.info('Start validation')
```

```
    groundtruth = []
    for i in range(test_steps):
        start = i* val_batch_size
        end = (i+1)*val_batch_size
        images_batch = []
        labels_batch = []
        labels_max_batch = []
        logger.info('=======start validation on {0}/{1} batch=========='.format(i, test_steps))
        for j in range(start,end):
            image_path = file_labels[j][0]
            temp_image = Image.open(image_path).convert('L')
            temp_image = temp_image.resize((FLAGS.image_size, FLAGS.image_size),Image.ANTIALIAS)
            temp_label = np.zeros([3755])
            label = int(file_labels[j][1])
            # print label
            temp_label[label] = 1
            # print "====",np.asarray(temp_image).shape
            labels_batch.append(temp_label)
            # print "====",np.asarray(temp_image).shape
            images_batch.append(np.asarray(temp_image)/255.0)
            labels_max_batch.append(label)
        # print images_batch
        images_batch = np.array(images_batch).reshape([-1, 64, 64, 1])
        labels_batch = np.array(labels_batch)
        batch_predict_val, batch_predict_index = sess.run([endpoints['predict_val_top3'],
                        endpoints['predict_index_top3']], feed_dict={images:images_batch, labels:labels_batch})
        logger.info('=======validation on {0}/{1} batch end=========='.format(i, test_steps))
        final_predict_val += batch_predict_val.tolist()
        final_predict_index += batch_predict_index.tolist()
        groundtruth += labels_max_batch
    sess.close()
    return final_predict_val, final_predict_index, groundtruth

```

在训练20w个step之后，大概能达到在测试集上能够达到：

```
val 980, final_predict_index [980, 1836, 2351]
val 2686, final_predict_index [2686, 2688, 2690]
val 1417, final_predict_index [1397, 1417, 1305]
eval on test dataset size: 223872
The accuracy 0.437223056032, the top3 accuracy 0.603134827044
ubuntu@10-19-168-132:/data/code/dl_opensource/toy_projects/chinese_rec/src$ vim train.py
```

相信如果在网络设计上多花点时间能够在一定程度上提升accuracy和top 3 accuracy.有兴趣的小伙伴们可以玩玩这个数据集。

Inference

```
def inference(image):
    temp_image = Image.open(image).convert('L')
    temp_image = temp_image.resize((FLAGS.image_size, FLAGS.image_size),Image.ANTIALIAS)
    sess = tf.Session()
    logger.info('========start inference============')
    images = tf.placeholder(dtype=tf.float32, shape=[None, 64, 64, 1])
    endpoints = network(images)
    saver = tf.train.Saver()
    ckpt = tf.train.latest_checkpoint(FLAGS.checkpoint_dir)
    if ckpt:
        saver.restore(sess, ckpt)
    predict_val, predict_index = sess.run([endpoints['predict_val_top3'],endpoints['predict_index_top3']], feed_dict={images:temp_im
    sess.close()
    return final_predict_val, final_predict_index

```

```
ubuntu@10-19-168-132:/data/code/dl_opensource/toy_projects/chinese_rec/src$ python train.py --mode=inference
I tensorflow/stream_executor/dso_loader.cc:128] successfully opened CUDA library libcublas.so locally
I tensorflow/stream_executor/dso_loader.cc:128] successfully opened CUDA library libcudnn.so.5 locally
I tensorflow/stream_executor/dso_loader.cc:128] successfully opened CUDA library libcufft.so locally
I tensorflow/stream_executor/dso_loader.cc:128] successfully opened CUDA library libcuda.so.1 locally
I tensorflow/stream_executor/dso_loader.cc:128] successfully opened CUDA library libcurand.so locally
inference
inference
I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one
I tensorflow/core/common_runtime/gpu/gpu_device.cc:885] Found device 0 with properties:
name: Tesla K80
major: 3 minor: 7 memoryClockRate (GHz) 0.8235
pciBusID 0000:00:08.0
Total memory: 11.25GiB
Free memory: 11.13GiB
I tensorflow/core/common_runtime/gpu/gpu_device.cc:906] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_device.cc:916] 0:   Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:975] Creating TensorFlow device (/gpu:0) -> (device: 0, name: Tesla K80, pci bus id: 0000:00:08.0)
========start inference============
the result info label 1066 predict index [[1066 1877 3613]] predict_val [[ 0.99194485  0.00312303  0.00183616]]
```

运气挺好，随便找了张图片就能准确识别出来

Summary

validation，inference，步骤都流程比较清晰， 美中不足的是，原本打算是在训练过程中，来对测试集做评估，但是在使用queue读test_data_dir下的filenames，和train本身的好像有点问题，不过应该是可以解决的，我这里就pass了。另外可能 还有一些可以改善的地方，比如感觉可以把batch data one hot的部分写入到network，这样，减缓在validation时内存会因为onehot的sparse开销比较大。

感觉这个中文手写汉字数据集价值很大，后面感觉会有好多可以玩的，比如

- 可以参考项亮大神的这篇文章端到端的OCR：验证码识别做定长的字符识别和不定长的字符识别，定长的基本原理是说，可以把最终输出扩展为k个输出， 每个值表示对应的字符label，这样cnn模型在feature extract之后就可以自己去识别对应字符而无需人工切割；而LSTM+CTC来解决不定长的验证码，类似于将音频解码为汉字
- 最近GAN特别火，感觉可以考虑用这个数据来做某个字的生成，和text2img那个项目text-to-image

这部分的代码都在我的github上tensorflow-101，有遇到相关功能,想参考代码的可以去上面找找，没准就能解决你们遇到的一些小问题.

Update in 2017.02.13

感谢@soloice的PR，使得代码更简洁， 并且修改了网络的结构，使得模型准确率上升很高, 最后top1和top3的结果：

```
the batch 1748 takes 0.0511560440063 seconds, accuracy = 0.8515625(top_1) 0.9
the batch 1749 takes 0.0512340068817 seconds, accuracy = 0.875(top_1) 0.96875
==================Validation Finished================
top 1 accuracy 0.829631547696 top k accuracy 0.928858748789
Write result into result.dict
Write file ends
```


相关文章

- 实现属于自己的TensorFlow(二) – 梯度计算与反向传播
- 15 分钟用 ML 破解一个验证码系统 · 12
- 实现属于自己的TensorFlow(1)：计算图与前向传播 · 1
- 图解机器学习：神经网络和 TensorFlow 的文本分类
- TensorFlow 实战：Neural Style

可能感兴趣的话题

- Eng --- React ,Vue ,Angula...
- 28岁刚入门，靠谱不？ · 8
- 工作四年，该如何晋级架构师？ · 1
- java 的nio和网络编程nio 的区别
- 工作压力大，然后出错多。。然后工作压力... · 6
- 是去北上广深闯几年还是留在二线城市？ · 2

最新评论

BurnellLiu （1 · ） 2017/03/21
程序员

感谢楼主，以前练习过MNIST数据集，一直没找到中文数据集，这次终于找到了。

赞 回复

王念一 （1 · ） 2017/04/01
高一学生

pangpangpangpangpangpangpang

赞 回复

羽恒 2017/08/14

感谢楼主，我可以对您的文章转载，或者借鉴您的文章写一些关于自己的理解吗

赞 回复

Python小组话题 我有新话题

Python学习，有哪些方向可以选择
小丑的哭笑 发起 • 30 回复

2年Java，想转 python
大概会吧 发起 • 16 回复

python 真的能在人工智能领域 一骑...
泽恒 发起 • 9 回复

零基础自学Python感觉很难，不像大...
keepcalm 发起 • 93 回复

有關 pandas 的問題
許秉凱 发起 • 2 回复

小弟机械行业3年了，自学了python半年，想...
阅微 发起 • 16 回复

- 本周热门Python文章
- 本月热门
- 热门标签

Python工具资源 更多资源 »

Tryton：一个通用商务框架
杂项

NLTK：一个先进的用来处理自然语言数据的Python程序。
自然语言处理 · 3

PyMC：马尔科夫链蒙特卡洛采样工具
科学计算与分析

statsmodels：统计建模和计量经济学
科学计算与分析

Pylearn2：一个基于Theano的机器学习库
机器学习 · 1

关于 Python 频道

Python频道分享 Python 开发技术、相关的行业动态。

快速链接
网站使用指南 »
加入我们 »
问题反馈与求助 »
网站积分规则 »
网站声望规则 »

关注我们

新浪微博：@Python开发者
RSS：订阅地址
推荐微信号

合作联系
Email：bd@Jobbole.com
QQ： 2302462408 （加好友请注明来意）

更多频道

小组 – 好的话题、有启发的回复、值得信赖的圈子
头条 – 分享和发现有价值的内容与观点
相亲 – 为IT单身男女服务的征婚传播平台
资源 – 优秀的工具资源导航
翻译 – 翻译传播优秀的外文文章
文章 – 国内外的精选文章
设计 – UI,网页，交互和用户体验
iOS – 专注iOS技术分享
安卓 – 专注Android技术分享
前端 – JavaScript, HTML5, CSS
Java – 专注Java技术分享
Python – 专注Python技术分享

 文章 小组 相亲 加入我们 反馈 沪ICP备14046347号-1